

N94-23959

OPERATIONAL CHARACTERISATION OF REQUIREMENTS AND EARLY VALIDATION ENVIRONMENT FOR HIGH DEMANDING SPACE SYSTEMS

E. Barro, A. Del Bufalo, F. Rossi

CISSET S.p.A.
Rome, Italy

ABSTRACT

The definition of some modern high demanding space systems requires a different approach to system definition and design from that adopted for traditional missions. System functionality is strongly coupled to the operational analysis, aimed at characterising the dynamic interactions of the flight element with its surrounding environment and its ground control segment. Unambiguous functional, operational and performance requirements are to be defined for the system, thus improving also the successive development stages. This paper proposes a Petri Nets based methodology and two related prototype applications (to ARISTOTELES orbit control and to Hermes telemetry generation) for the operational analysis of space systems through the dynamic modelling of their functions and a related computer aided environment (ISIDE) able to make the dynamic model work, thus enabling an early validation of the system functional representation, and to provide a structured system requirements data base, which is the shared knowledge base interconnecting static and dynamic applications, fully traceable with the models and interfaceable with the external world.

Key Words: System Operations, Functional Analysis, Dynamic Simulation, Validation, Petri Nets.

1. OPERATIONAL FEATURES OF HIGH DEMANDING SPACE MISSIONS

Some of the planned European space missions (Low Earth Orbit or Deep Space missions, like ARISTOTELES, Cassini, Mars mission) present a significant increase of complexity in the spacecraft system definition with respect to the traditional communications or scientific satellites.

The main reason is that their specific operational constraints (short and rare ground contact periods, or decreasing ground control capabilities and performances due to long distances) have for this kind of missions a stronger impact on system architecture.

Such constraints impose the need to define a high degree of on board autonomy for the spacecraft, or, in other words, to identify specific operations driven flight element control functions which in the

traditional missions were typically allocated to the ground segment, finding a good compromise between cost and complexity of a self standing system and the operational risks associated with the delegation of tasks.

In addition, for these missions the system operator can rely on a limited budget of information about the spacecraft, which must be carefully defined in order to ensure the safety of the spacecraft and to optimise both the system monitor and control loop and the payload exploitation.

Such kind of problems can never be solved only on the basis of a previously consolidated experience in "similar" past missions, as the definition of system autonomy and the consequent spacecraft design are heavily constrained by the dynamic interactions of the flight element with its surrounding environment and its ground control segment, which are strictly mission specific.

2. AN ALTERNATIVE APPROACH TO SYSTEM DEFINITION AND OPERATIONS DESIGN

As a matter of fact, the correctness in the identification of the optimum sharing of functions between on board and ground is critical for the definition of suitable functional and performance requirements for the spacecraft, which are the baseline for the system architecture.

As a consequence, an in depth analysis of the operations related aspects of modern spacecrafts (and therefore of the system dynamic behaviour) is fundamental even in the early system functional analysis and requirement specification phase of the spacecraft development.

Due to the relevance of the spacecraft operational aspects in the definition of the system architecture, it is very important in this phase to demonstrate that the system built by the designer is capable of fulfil in its working environment the identified functionality, especially if the system functioning is subject to severe time constraints.

The current analysis methodologies used for the system definition phase take only partially into account the operational aspects, and ensure only a preliminary coherence of the system functional model with the derived requirements.

In this phase usually the designer:

- o establishes some operational choices for the system, taking into account the constraints imposed by the context;
- o identifies a hierarchical structure of system functions, together with their relevant attributes;
- o formalises the system functions into a set of functional and performance requirements;
- o builds the first architectural design of the system, where the system requirements are translated into a physical architecture, on the basis of the a priori implementation constraints imposed by the user.

The functional model, however, is not currently able to provide an exhaustive representation of the system, as the usual modelling methodologies (e.g. SADT, OODLE) are all static.

The system dynamics, i.e. the representation of its dynamic behaviour and the modelling of the system operations is generally not taken into account in the system definition phase.

As a consequence the specification of dynamic requirements for the system is not usually derived from the characteristics of the model.

This lack in the system definition suggests the need to introduce a more complete and consistent approach to this phase, in order to tightly link the flight element functions and the related static and dynamic requirements to the operations concept identified for the spacecraft, ensuring their full consistency.

This approach can be enforced by exploiting an support environment aimed at providing the system designer with aids for the generation of a complete and fully traceable model of the system at functional level, by means of:

- o modelling the system static and dynamic behaviour;
- o building a coherent and consistent set of system requirements;
- o validating the model versus the system requirements and the operational strategies;
- o providing a significant control over the next steps of system life cycle (system architectural design).

Such a support environment, named Integral System Investigation and Definition Environment (ISIDE), is currently being developed by Ciset; its basic principles and features are described in section 5.

3. A TYPICAL EXAMPLE: THE ARISTOTELES MISSION

A subset prototype of ISIDE, the System Dynamic Analysis Environment, has been developed and successfully used in the field of spacecraft operations analysis in the frame of ARISTOTELES phase pre-B

studies, related to satellite autonomy concept definition (Ref. 2, 3).

Such a prototype provides the capability to build and execute a dynamic executable functional representation of the system (based on the well known Petri Nets methodology, Ref. 1) or of an operational process.

The representation is parametrised by means of a direct link with a System Requirements Data Base.

The objective is to verify the validity of operations concepts with respect to system requirements and to the spacecraft operational context.

The representation (model) can then be run as a real simulator, providing as an output statistical figures of the system dynamics and enabling the assessment of the correctness of the tested operational approach and the identification of critical paths in the process execution (bottlenecks, deadlocks).

The basic steps of model generation are:

- o the building and parametrisation of a dynamic functional representation of the process (built with Petri Nets methodology): the representation is obtained from a static SADT model by means of translation rules and by adding to the static model the system dynamic information (missing in SADT) as derived from the Requirements DB.
- o the execution of an ad hoc simulation, the output of which enables the validation of the concept under study and the generation of statistical information on the model behaviour.

3.1 Description of the model

The prototype has been extensively used for the analysis and early validation of a proposed operational concept of ARISTOTELES, a very low earth orbit satellite ($h=200$ Km, $i=98.5^\circ$), hosting a gravity gradiometer aimed at an accurate measurement of the earth gravity field.

The strategy for keeping the satellite inside its allowed deadband (± 3 Km), through the execution of dedicated Orbit Raise Manoeuvres (ORM), was critical due to reduced GS coverage (only Kiruna ground station available), limited S/C weight and high decay rate (about .240 m/h)

Further constraints came from the S/C critical safety conditions and the limited accuracy of orbit determination process during specific contingency situations.

The autonomy concept definition, therefore, focused on the splitting of orbit control process functions between the ground and the space segment.

The basic choice was thus represented by identifying where to perform the computation of the predicted times and amount of orbit raise manoeuvres, combining it with a suitable operational strategy and with the constraints coming from the environment.

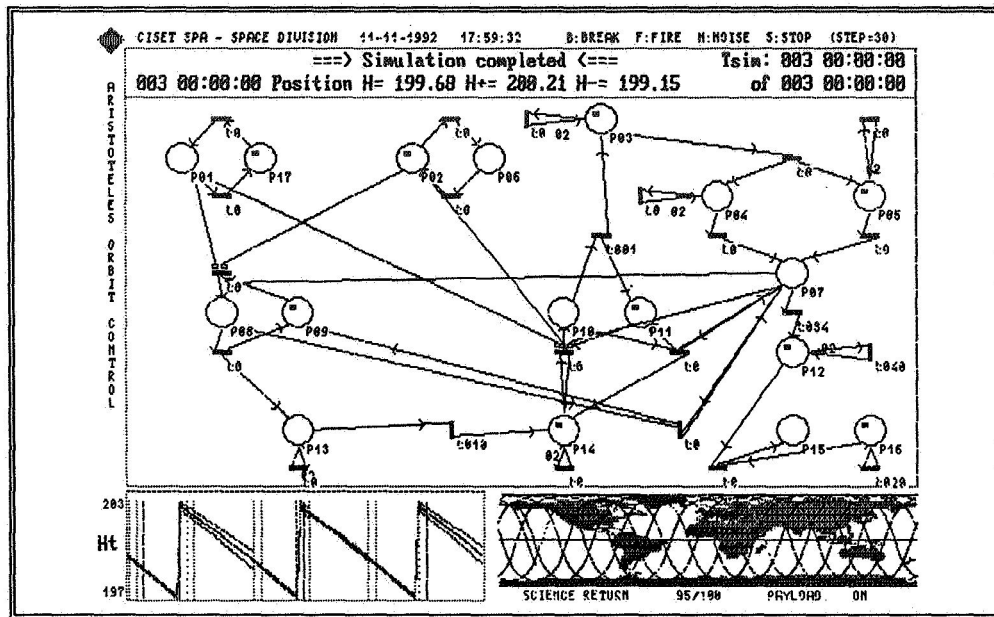


Fig. 1: ARISTOTELES Orbit Raise Manoeuvre Model simulation output.

A dynamic model of the ORM was built and put at work. In figure 1 the final graphical output of a 3 days model simulation is presented. The following table shows the significance of the network places.

1	Ground Contact enabled (no failure)
2	Kiruna is visible
3	Next Manoeuvres times on board
4	Man. n ready for execution
5	Man. n+1 ready for execution
6	Kiruna is not visible
7	Manoeuvre in execution
8	Down link in execution
9	Down link standby
10	Up link in execution
11	Up link standby
12	Manoeuvre successfully executed
13	Satellite Tracking executed
14	Next Man. times computed on ground
15	Satellite out of deadband
16	Satellite within the deadband
17	Ground Contact inhibited (failure)

Table 1: ORM Petri Net places description.

The model is based on a Petri Nets 'engine' describing the overall functional mechanism of the process, including ground functions (tracking, manoeuvres times computation and up-link), spacecraft functions (manoeuvres execution, down-link) and the spacecraft environment influence on the process (ground station visibility, contact failures), which schedules the various simulation modules.

Two major external simulators are interfaced:

- o an orbital propagator, driving the spacecraft visibility on the basis of its initial position, orbital parameters and of Kiruna features;
- o an atmospheric drag model, which computes the satellite altitude (including altitude determination

errors) on the basis of a drag simulation algorithm, taking as an input from the network the manoeuvres executed and releasing as an output the current satellite altitude.

The Petri network is parametrised with the dynamic information about the process (e.g. altitude deadband, characteristic times) which are derived a priori from a database of system requirements. The graphical display, as shown, combines the Net with the output of the two simulators (on the bottom).

3.2 Simulation results

The simulation of ORM process for different initial conditions and environmental conditions enabled the validation of the tested operations strategy, once fixed the value of system parameters provided within the system requirements database.

Furthermore it allowed to verify the sensitivity of the strategy to the variation of any of the parameters of the model.

Finally, the simulation execution provided a wide number of statistical results about the process under study, like the distribution of manoeuvres intervals and of manoeuvres size, the deadband utilisation figure, the scientific return comparing those values with the expectations at System Requirements level.

4. CHARACTERISTICS OF THE PROTOTYPE MODELLING ENVIRONMENT

The System Dynamic Analysis Environment used for ARISTOTELES ORM model was developed on IBM PS2 using C language under DOS 5.0.

The prototype architecture, as shown in figure 2, assembles three separate environments:

- o a system modelling environment (PN, SRDB, simulation modules and link editors);
- o a simulation execution environment;
- o an evaluation environment.

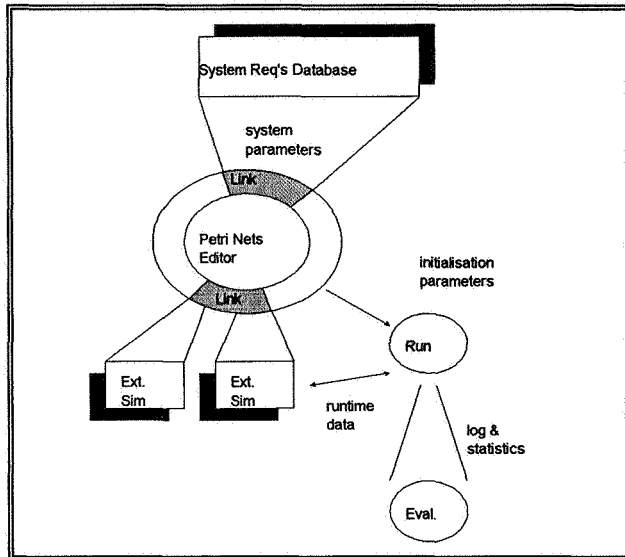


Fig. 2: Modelling environment architecture.

The model preparation is based on a Petri Nets Editor with the following characteristics associated with the network transitions:

- o multiple and inhibitor arcs;
- o deterministic firing time;
- o firing conditions (including random);
- o actions executed on transitions firing (e.g. activation of simulation modules).

4.1 Model preparation and system requirements

By means of this editor the system engineer can build the network which models the process (or the system) under study, defining the process mechanism and the related transition characteristics (firing time, conditions, actions), and identifying the set of data, variables or commands which constitute the interface of the model with any external software (e.g. an external simulator).

The editor also enables the creation of a link of variables with a **System Requirements Database**, which can be generated and maintained separately by means of a database editor.

Whenever the database information is changed, the network parameters used for the simulation run are updated accordingly.

4.2 Simulation Execution

Once the model has been generated, a simulation can be executed by means of the run-time module.

All the model parameters derived from the system requirements database can be accepted or modified in this phase. In addition, other simulation initialisation parameters, like simulation time step can be set.

The run-time module executes the simulation according to the Petri Nets syntax, invoking external simulation modules for conditions verification and actions performing. The capability of defining firing conditions for the network transitions enables the implementation of functional priorities, in case the modelled process is fully deterministic (no resource conflict between concurrent functions is allowed).

The definition of transitions associated actions enables the parametrisation of network tokens, modelling in this way the availability of different kind of resources within the system.

All the significant simulation events and parameters (transitions firing, parameters values) are displayed and logged. The display messages can be defined in a customised way during the model preparation, and may include the monitor current values of model internal and external variables.

4.3 Simulation Evaluation

After the simulation execution, the log file is processed by an **Evaluation** module, which computes and displays the main network statistics, i.e. for each transition:

- o overall number of firings;
- o minimum, average and maximum time between two successive firings.
- o pre-defined statistical figures of selected network parameters.

The module also allows the navigation within the log file (e.g. searching for all the occurrence of a pre-defined event).

5. INTEGRAL SYSTEM IDENTIFICATION AND DEFINITION ENVIRONMENT (ISIDE)

The above described System Dynamic Analysis Environment is a preliminary application of a more general concept, ISIDE.

ISIDE is aimed at providing a computer aided environment for the generation of an integral and consistent system description and for its validation in the frame of the system definition phase.

The fundamental idea behind ISIDE is the integration of a functional static and dynamic representation of the system and its specification into a set of system requirements, addressing an integral system model, where all the system related information are coherently collected.

From the ISIDE viewpoint the system requirements have not to be considered as a further information of

the system, but they grow up together with the functional and dynamic models, being strictly linked to them by means of the ISIDE syntax.

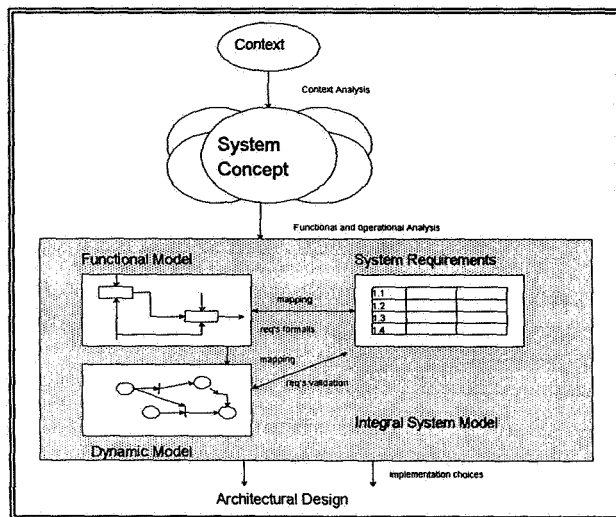


Fig. 3: ISIDE Concept.

The models enable the early verification of the correctness of the system concept, and as a consequence of the fitness of the model to the user needs. In addition, the possibility of describing the system dynamics enables even in the architectural design phase the verification at a functional level of the choices made, in terms of system functioning.

5.1 Summary of ISIDE Features

ISIDE will provide the following capabilities:

- 1) Modelling the system static and dynamic behaviour.
ISIDE defines a rigorous syntactic link between the static and dynamic models, in order to ensure they describe exactly the same system (the dynamic model is "automatically" derived from the functional model).
- 2) Building a coherent and consistent set of system requirements.
The system requirements are considered in the context of ISIDE the core of the system representation. They are structured, parametrised and directly interfaced with the entities and parameters of the models.
This ensures traceability with the models, providing full flexibility of the system representation.
- 3) Validating the model.
The dynamic model provides the capability to show, via an executable simulation, that the system works, at functional level, in compliance with user needs, time constraints and operational choices reflected in the system concept.
- 4) Controlling system life cycle.

The representation capability of the dynamic model can also be exploited in the next phases of system life cycle, where the model can be easily enriched with additional parameters coming from the implementation choices.

From the point of view of ISIDE implementation, the environment is built by means of the integration of:

- o an SADT functional modelling tool;
- o a Petri Net dynamic modelling tool;
- o an ORACLE based system requirements DB.

The use of ORACLE and C based interfaces ensures ISIDE will be fully open to the external world, thus enabling a wide utilisation of ISIDE (e.g. integration with external simulators, use of the database along different phases of the system life-cycle) as specific functional or data interface may be set for the exchange of relevant parameters.

On the other hand, the system knowledge base may also be easily maintained, evolving in the various phases of system definition, up to becoming a real operational database of the system.

5.2 Application of ISIDE to Architectural Design

As already outlined, although the ISIDE concept was born for covering lacks in the early system definition phase, the environment characteristics make it effective to exploit ISIDE also during the successive phases of the system life cycle, and in particular during the architectural design phase. That is natural when considering that the hierarchical nature of the methodologies used for system description (SADT and Petri Nets) enables the progressive detailing of the model in parallel with the system evolution.

As an example of an application of ISIDE to system design we propose a model, generated and executed using the System Dynamic Analysis Environment, of a software for the simulation of Hermes on board telemetry generation process.

The actual software system, under development by Ciset in the frame of the Board Observability Breadboard project within the Hermes Programme (Ref. 4), will generate in real time telemetry packets filled with measurement values varying according to pre-defined variation laws. The telemetry generator is interactively commanded by test operator directives issued according to a telemetry plan, and sends the generated packets to communications simulator for space to ground link modelling. The simulator implements on board recording and packets playback functions, together with the filling of high rate telemetry with dummy packets when required.

The aim is to validate the software functional specifications with respect to the identified priorities for the S/W processes and to assess the overall system performances on the basis of the times needed for the execution of each elementary task.

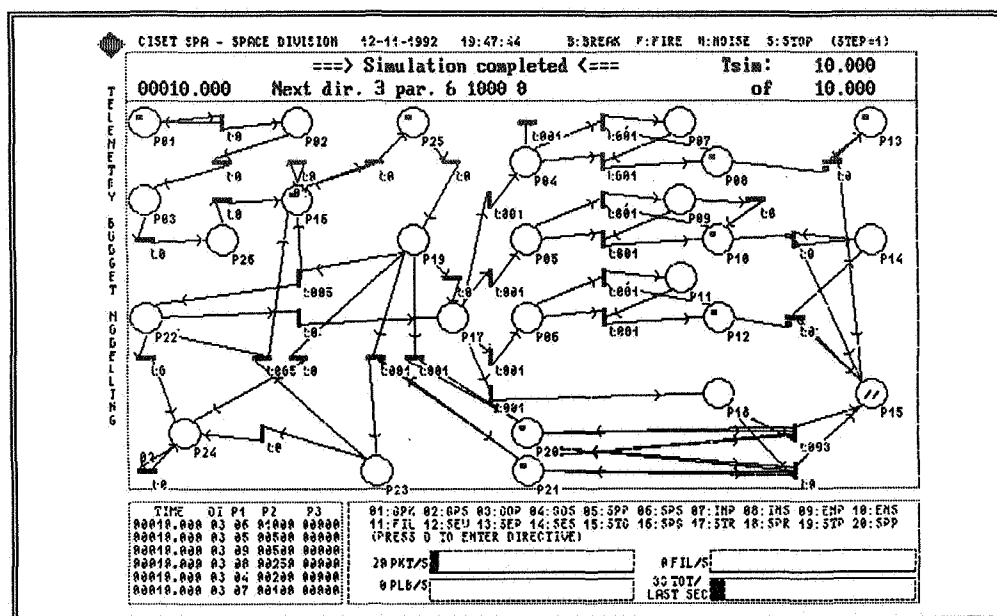


Fig. 4: Telemetry Generation Software model simulation output.

The model of the telemetry generation simulation software is shown in figure 4, the significance of the places of the network is described in table 2.

1	Directive Input Enabled
2	Directive acquired
3	Directive Translated
4	Packet filler command in execution
5	Playback command in execution
6	Recorder command in execution
7	Packet filler on
8	Packet filler off
9	Playback on
10	Playback off
11	Recorder on
12	Recorder off
13	Filling Packets file ready for filling
14	Recorded Packets file ready for playback
15	Generated Packets ready for delivery
16	Directive Scheduled
17	Next command in execution
18	Packet Generation Command in execution
19	Directive Interpreted
20	Packet File Updated
21	Measurements File Updated
22	Directive rejected
23	Directive rescheduled for execution
24	End of directive processing
25	Read next directive from schedule
26	Directive Accepted

Table 2: TM Generator Petri Net places description.

6. CONCLUSIONS

The main advantages introduced by ISIDE are:

- o a wider , more rigorous and 'operations driven' description of the system;
- o an early assessment of system correctness;
- o a re-use of existing simulators with easy upgrade.

On the other hand, the possible drawbacks are:

- o its initial costs;
- o it requires training and workstations;
- o its utilisation could be too time and manpower consuming.

These drawbacks turn out to be not significant, when considering that:

- o the initial cost increase for system definition (tools, training, hardware, manpower) will certainly bring a much more consistent cost decrease in the next phases of system life cycle, due to the possibility of detecting and solving design errors in an earlier stage of the project;
- o the characteristic of ISIDE to be an open environment enables the maintenance and further exploitation of its products throughout the whole system life cycle.

7. REFERENCES

1. Agerwala T. 1979. Putting Petri Nets to work. *Computer*, Dec. '89, 85-94
2. CISSET. ARISTOTELES Phase Pre-B Study: Satellite Autonomy. Ref. ARI-TN-CI-001, Issue 1.1, April 3rd, 1992
3. E. Barro & F. Rossi. An Application of Timed Petri Nets to Operations Analysis: the ARISTOTELES Autonomy Concept. In *Proc. ESA Symp. "Ground Data Systems for Spacecraft Control"*, Darmstadt, FRG, ESA SP-308, 317-322.
4. SAT CONTROL. BOB Software and Hardware Architecture Description. H-NT-01210-0286-SATC, Issue V0, September 8th, 1992.